REMARKS

Applicant respectfully requests reconsideration and allowance of all of the claims of the application. The status of the claims is as follows:

- Claims 1, 3, 4, 6-12, 14, 17-19, 21, and 23-31 are currently pending.
- Claims 1, 12, and 19 are amended herein.

Support for the claim amendments is found in the specification, as originally filed, at least at pages 12-14 and Figure 5. The amendments submitted herein do not introduce new matter.

Cited Documents

The following documents have been applied to reject one or more claims of the Application:

- **Dussud**: Dussud, U.S. Patent No. 6,502,111
- **Paragraph [0007]**: Paragraph [0007] of the Background of the Specification

Claims 1, 3, 4, 6-12, 14, 17-19, 21, and 23-31 are Non-Obvious Over Dussud in view of Paragraph [0007]

Claims 1, 3, 4, 6-12, 14, 17-19, 21, and 23-31 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Dussud in view of Paragraph [0007]. Applicant respectfully traverses the rejection. Nevertheless, solely in the interest of expediting issuance, Applicant amends the claims as shown above to highlight claimed distinctions as discussed below. Applicant respectfully requests reconsideration in light of the amendments presented herein.

Independent Claim 1

Claim 1, as amended herein, recites, in part:

A computer-readable storage medium apparatus having computer-executable instructions encoded thereon to support ephemeral garbage collection by setting a write-watch mechanism to watch specified memory locations, the computer-readable storage medium apparatus being accessible by a computing device, the instructions when executed, configuring the computing device such that during execution of a program, when a statement of the program for execution is obtained, the computing device is configured to determine whether the statement includes a store operator, the computing device being further configured to perform operations comprising:

during a loop that is performed for at least two iterations based at least on respective store operators:

storing a value specified in the statement in a memory location specified in the statement;

determining whether the memory location specified is within an ephemeral generation;

in an event the memory location specified is within the ephemeral generation, obtaining a next statement of the program for execution; and

in an event the memory location specified is not within the ephemeral generation, setting a card associated with the memory location specified and obtaining the next statement of the program for execution;

requesting via the write-watch mechanism a list of memory locations, the list:

identifying a plurality of the memory locations that have been accessed since a last ephemeral garbage collection process, each memory location corresponding to one of a plurality of cards associated with one or more objects allocated from within a memory heap, each of the plurality of cards associated with a card table, wherein the card table identifies one or more of the plurality of cards with objects that have been accessed; and

comprising a bitmap, wherein each bit within the bitmap corresponds to one of the plurality of cards, modification of the bitmap occurring when a corresponding bit is set at the time that the card is trimmed to disk;

creating, during the current ephemeral garbage collection process and responsive to exiting the loop upon encountering a statement lacking a store operator, a bundle table containing entries identifying a plurality of bundles, wherein each of the plurality of bundles identifies groupings of subsets of the plurality of cards;

marking, outside of the loop including setting the card, during the current ephemeral garbage collection process, two or more of the plurality of bundles identified in the bundle table using the list, wherein the marked bundles identify groupings of subsets of the plurality of marked cards having associated objects that have been accessed since the last ephemeral garbage collection process; and

performing garbage collection upon at least one accessed object.

Dussud is directed to concurrent garbage collection and describes that "modifications in the memory structure that occur during the concurrent marking act are logged ... by a write watch module. The application is then paused or stopped to perform a second marking act using information from the write watch module." *Dussud*, Abstract.

Cited paragraph [0007] discusses that "during garbage collection, the bundle bit map is checked to determine which bundles have objects that have been accessed. If the bundle bit map indicates that an object within the bundle has been accessed, each card in that bundle is checked to see if it has been accessed. If it has, then each of its objects is checked. While this technique improves the efficiency of the ephemeral garbage collection, the cost of executing the program is doubled. ... [T]he addition of [a] statement to perform bundling doubles the overhead for executing [a] store operator ... in comparison to only performing card-marking." *Paragraph [0007]*, Application, p. 3.

Thus, Dussud describes concurrent garbage collection with two instances of marking, and paragraph [0007] discusses the problem of doubled overhead associated with two instances of marking. The purported combination of Dussud and paragraph

[0007] would merely exemplify the problem described in paragraph [0007]. In contrast, the claim removes a second instance of marking to an occurrence outside the loop so that the second instance of marking need not be concurrent.

As discussed during the interview, a loop as presented in the originally filed specification including the figures and as claimed would be understood by one of ordinary skill at the time of the filing of the instant application to support the instructions of the loop being performed more than once. Otherwise, such operations would merely represent a series of steps, without paths providing that the loop would be performed more than one time. Nevertheless, claim 1 is amended to emphasize instructions of the claimed loop being performed at least twice based at least upon executing a store operator solely in the interest of highlighting that the claimed loop is indeed a loop, and not a series of instructions performed but once as broadly interpreted in the Office Action. *See Action,* p. 19.

Moreover, neither Dussud nor paragraph [0007], alone or in combination, have been shown to teach or suggest at least "determin[ing] whether [a program] statement includes a store operator [and] during a loop that is performed for at least two iterations based at least on respective store operators: setting a card associated with [a specified] memory location," "creating, during the current ephemeral garbage collection process and responsive to exiting the loop upon encountering a statement lacking a store operator, a bundle table," and marking, outside of the loop including setting the card, ... two or more of the plurality of bundles identified in the bundle table using the list," as recited in amended claim 1.

Consequently, the combination of Dussud and paragraph [0007] does not teach or suggest at least this element of claim 1.

For at least the reasons presented herein, the combination of Dussud and paragraph [0007] does not teach or suggest all of the features of claim 1. Accordingly, Applicant respectfully requests that the Office withdraw the § 103 rejection of claim 1.

Dependent Claims 3, 4, 6-11, 28, and 29

Claims 3, 4, 6-11, 28 and 29 ultimately depend from independent claim 1. As discussed above, claim 1 is allowable over the cited documents. Therefore, claims 3, 4, 6-11, 28 and 29 are allowable over the cited documents of record for at least their dependency from an allowable base claim, and for the additional features that each recites.

For example, claim 6 recites that "the write-watch mechanism maintains the list of memory locations in response to a request from the ephemeral garbage collection process." In rejecting claim 6, the Office cites Dussud. *Action*, p. 7. The cited portion of Dussud states that "the garbage collector 30 requests the information accumulated by the write watch module 32." *Dussud*, c. 5, ll. 50-51. Thus, Dussud discusses that the garbage collector requests information from the write watch module. However, Dussud has not been shown to teach or suggest the garbage collector requesting that the write watch module maintain the information as recited in claim 6.

Accordingly, Applicant respectfully requests that the Office withdraw the 103 rejection of claims 3, 4, 6-11, 28 and 29.

Independent Claim 12

Claim 12, as amended herein, recites, in part:

> A method for executing statements within a program to support ephemeral garbage collection by setting a write-watch mechanism to watch specified memory locations such that during execution of a program, when a statement of the program for execution including a store operator is obtained, a computing device is configured to perform the method comprising:
>
> specifying a range of card table memory to watch during program execution by calling a write-watch mechanism that:
>
> performs tracking of access to the card table memory; and
>
> maintains a write-watch list that identifies cards marked within the card table memory since a garbage collection process was last performed, each card being associated with and updated upon access to one or more objects allocated within a memory heap, the memory heap being divided into a plurality of cards with each card being grouped into one of a plurality of bundles, wherein one of the plurality of bundles corresponds to a subset of that plurality of cards that are tracked using a page of card table memory outside of a loop that is traversed at least twice based on a respective number of store operators including marking at least one of the plurality of cards;
>
> during the loop including marking at least one of the plurality of cards, in an event the statement obtained has one of the store operators:
>
> storing a value within the memory heap at a memory location specified by the statement obtained;
>
> determining whether the memory location specified is within an ephemeral generation;
>
> in an event the memory location specified is within the ephemeral generation, obtaining a next statement of the program for execution; and
>
> in an event the memory location specified is not within the ephemeral generation, marking the at least one of the plurality of cards within the card table memory corresponding to the memory location and obtaining a next statement of the program for execution.

As discussed above, Dussud describes concurrent garbage collection with two instances of marking, and paragraph [0007] discusses the problem of doubled overhead

associated with two instances of marking. In contrast, the claim removes a second instance of marking to an occurrence outside the loop so that the second instance of marking need not be concurrent. At least for similar reasons as those discussed above, the purported combination of Dussud and paragraph [0007] would merely exemplify the problem described in paragraph [0007].

However, neither Dussud nor paragraph [0007], alone or in combination, have been shown to teach or suggest at least that "one of [a] plurality of bundles [of grouped cards] corresponds to a subset of that plurality of cards that are tracked ... outside of a loop that is traversed at least twice based on a respective number of store operators including marking at least one of the plurality of cards; [and] during the loop including marking at least one of the plurality of cards in an event the statement obtained has one of the store operators, ... in an event the memory location specified is not within the ephemeral generation, marking the at least one of the plurality of cards ... and obtaining a next statement of the program for execution," as recited in amended claim 12.

Consequently, the combination of Dussud and paragraph [0007] does not teach or suggest at least this element of claim 12.

For at least the reasons presented herein, the combination of Dussud and paragraph [0007] does not teach or suggest all of the features of claim 12. Accordingly, Applicant respectfully requests that the Office withdraw the 103 rejection of claim 12.

Dependent Claims 14, 17, 18, 25, 26, and 30

Claims 14, 17, 18, 25, 26, and 30 ultimately depend from independent claim 12. As discussed above, claim 12 is allowable over the cited documents. Therefore, claims 14, 17, 18, 25, 26, and 30 are allowable over the cited documents of record for at least

their dependency from an allowable base claim. Accordingly, Applicant respectfully requests that the Office withdraw the 103 rejection of claims 14, 17, 18, 25, 26, and 30.

Independent Claim 19

Claim 19, as amended herein, recites:

> A memory management system configured to set a write-watch mechanism to watch specified memory locations during execution of a program, obtain a statement of the program for execution, and determine whether the statement obtained includes a store operator, the system comprising:
>
> a processor;
>
> a memory into which a plurality of instructions are loaded and into which a plurality of objects are dynamically allocated, the memory having a heap into which the objects are allocated, the heap being divided into a plurality of cards which are grouped into a plurality of bundles, each card being associated with one or more of the plurality of objects, wherein upon execution of the plurality of instructions by the processor, the system being configured to, based at least on whether the store operator is included in the statement for execution obtained, perform an operation such that:
>
> in an event the statement obtained does not have a store operator, executing the statement; and
>
> in an event the statement obtained has a store operator performing at least two iterations of a loop including:
>
> storing a value specified in the statement obtained in a memory location specified in the statement obtained;
>
> determining whether the memory location specified is within an ephemeral generation;
>
> in an event the memory location specified is within the ephemeral generation, obtaining a next statement of the program for execution; and
>
> in an event the memory location specified is not within the ephemeral generation, setting a card associated with the memory location specified and obtaining the next statement of the program for execution; and
>
> the write-watch mechanism configured to identify cards that have been set in the memory location specified since a garbage collection process was last performed, the plurality of cards being grouped into one

of the plurality of bundles, and a corresponding bundle of the plurality of bundles that have been marked outside of the loop based at least on encountering a statement not having a store operator including setting the card.

As discussed above, Dussud describes concurrent garbage collection with two instances of marking, and paragraph [0007] discusses the problem of doubled overhead associated with two instances of marking. In contrast, the claim removes a second instance of marking to an occurrence outside the loop so that the second instance of marking need not be concurrent. At least for similar reasons as those discussed above, the purported combination of Dussud and paragraph [0007] would merely exemplify the problem described in paragraph [0007].

However, neither Dussud nor paragraph [0007], alone or in combination, have been shown to teach or suggest at least a system "configured to ... based at least on whether [a] store operator is included [in a statement] perform[] a loop [that is performed for at least two iterations, the loop] including: setting a card associated with the memory location specified ... and [the system including a] write-watch mechanism [that is] configured to identify cards that have been set[, a] plurality of cards being grouped into one of [a] plurality of bundles, and a corresponding bundle of the plurality of bundles that have been marked outside of the loop based at least on encountering a statement not having a store operator including setting the card," as recited in amended claim 19.

Consequently, the combination of Dussud and paragraph [0007] does not teach or suggest at least this element of claim 19.

For at least the reasons presented herein, the combination of Dussud and paragraph [0007] does not teach or suggest all of the features of claim 19. Accordingly, Applicant respectfully requests that the Office withdraw the 103 rejection of claim 19.

Dependent Claims 21, 23, 24, and 31

Claims 21, 23, 24, and 31 ultimately depend from independent claim 19. As discussed above, claim 19 is allowable over the cited documents. Therefore, claims 21, 23, 24, and 31 are allowable over the cited documents of record for at least their dependency from an allowable base claim. Accordingly, Applicant respectfully requests that the Office withdraw the 103 rejection of claims 21, 23, 24, and 31.

Conclusion

For at least the foregoing reasons, all pending claims are in condition for allowance. Applicant respectfully requests reconsideration and prompt issuance of the application.

If any issues remain that would prevent allowance of this application, **Applicant requests that the Examiner contact the undersigned attorney before issuing a subsequent Action.**

Respectfully Submitted,

Lee & Hayes, PLLC
Representatives for Applicant

By: /Bea Koempel-Thomas 58213/ Dated: February 24, 2011

Beatrice L. Koempel-Thomas
(bea@leehayes.com; 509-944-4759)
Registration No. 58213

Lewis C. Lee
(lewis@leehayes.com; 509-944-4711)
Registration No. 34656